Polynomial Time Inference of Extended Regular Pattern Languages

AUTHOR(S):

Shinohara, Takeshi

CITATION:

Shinohara, Takeshi. Polynomial Time Inference of Extended Regular Pattern Languages. 数理解析研究所講究録 1983, 482: 115-128

ISSUE DATE:

1983-03

URL:

http://hdl.handle.net/2433/103408

数理解析研究所講究録
482巻 1983年 115-128

Polynomial Time Inference of Extended Regular Pattern Languages

Takeshi Shinohara

Computer Center, Kyushu University 91,
Fukuoka, 812 Japan

ABSTRACT

A pattern is a string of constant symbols and variable symbols. The language of a pattern p is the set of all strings obtained by substituting any non-empty constant string for each variable symbol in p. A regular pattern has at most one occurrence of each variable symbol. The class of pattern languages was introduced and discussed by Angluin[2]. In the previous paper [Shinohara, 9] we have studied polynomial time inference from positive data about the class of regular pattern languages.

In this paper we consider this problem in case of extended regular pattern languages which are sets of all strings obtained by substituting any (possibly empty) constant string instead of non-empty string. Our inference procedure uses MINL calculation, introduced by Angluin [2], which finds a minimal language containing a given finite set of strings. The relation between MINL calculation for the class of extended regular pattern languages and the longest common subsequence problem is also discussed.

1. Introduction

There have been known two kinds of inferences, deductive inference and inductive inference. Many studies on deductive inference cover a wide range from theoretical problems to practical problems. Although some theories of inductive inference have been developed, few of them have reached practical applications to computer softwares. This study presents an approach to practical applications of inductive inference and give its theoretical basis.

Inductive inference of languages, we consider in this paper, is called "polynomial time inference from positive data." The direct motivation of this research is to develop a data entry system with learning function, proposed by Arikawa[4]. The system must infer or learn the structure of input data from the user. The information, the system can use, is only the input data. Hence we should consider inference from positive data. The computational complexity problem is another important point in discussing such practical problems as the learning data entry system. Polynomial time inference is an inference carried out by a machine which makes every guess in polynomial time.

It has been considered of little interest to study inference from positive data, since Gold[5] proved a strong theorem which asserts that any class of languages over an alphabet is not inferrable from positive data if it contains all finite languages and at least one infinite language. Hence, for example, the class of regular sets is not inferrable from positive data. Recently Angluin[2,3] gave new life to the study of inference by characterizing the class of languages inferrable from positive data and presenting interesting classes. The class of pattern languages is one of her classes.

A pattern is a string of constant symbols and variable symbols, and the language of a pattern p is the set of all strings obtained by substituting any non-empty constant string for each variable symbol in p. Shinohara[9] has shown that two subclasses of pattern languages, named regular pattern languages and non-cross pattern languages, are polynomial time inferrable from positive data. A regular pattern is a pattern in which each variable symbol occurs at most once.

In this paper we first point out some problems of our previous version of inference method and then we give a solution to them by considering polynomial time inferrability of extended regular pattern languages. Our extension is to allow the substitutions to erase some variable symbols. For example, the extended language of a pattern 0x1 can contain string "01" while the language by Angluin can not. The erasing variables requires a new discussion.

The inference, we deal with here, is carried out by using MINL calculation introduced by Angluin[2]. Hence our main attention is paid to the time complexity of MINL calculation for the class of extended regular pattern languages. MINL for extended regular pattern languages finds a regular pattern which represents a minimal extended regular pattern language containing a given non-empty finite set of strings. We also refer to the relation between MINL calculation for extended regular pattern languages and the longest common subsequence problem. We propose an algorithm which calculates MINL for extended regular pattern languages in polynomial time. By using this fact, the class of extended regular pattern languages is shown to be polynomial time inferrable from positive data.

2. Preliminaries

We begin with a brief review of our previous results.

2.1. Patterns and Their Languages

Let Σ be a finite set of symbols containing at least two symbols and let $X = \{x_1, x_2, \ldots\}$ be a countable set of symbols disjoint from Σ. Elements in Σ are called constants and elements in X are called variables. A pattern is any string over $\Sigma \cup X$. The set $(\Sigma \cup X)^*$ of all patterns is denoted by P.

We say that a pattern p is regular if each variable in p occurs exactly once in p.

Let f be a non-erasing homomorphism from P to P. If $f(a) = a$ for any constant a, then f is called a substitution. If f is a substitution, $f(x)$ is in X, and $f(x) = f(y)$ implies $x = y$ for any variables x and y, then f is called a renaming of variables. We use a notation $[a_1/v_1, \ldots, a_k/v_k]$ for the substitution which maps each variable symbol v_i to a_i and every other symbol to itself. We define two binary relations on P as follows:

1) $p \equiv' q$ iff $p = f(q)$ for some renaming of variables f,
2) $p \leq' q$ iff $p = f(q)$ for some substitution f.

The language of a pattern p, denoted by $L(p)$, is the set $\{ w \in \Sigma^* \mid w \leq' p\}$. These syntactic relations $\equiv'$ and $\leq'$ are characterized by the following lemma.

Lemma 1. [Angluin, 2]

1) For all patterns p and q, $p \equiv' q$ iff $L(p) = L(q)$.

2) For all patterns p and q, if $p \leq' q$ then $L(p) \subseteq L(q)$, but the converse is not true in general.

3) If p and q are patterns such that $|p| = |q|$, then $p \leq' q$ iff $L(p) \subseteq L(q)$.

2.2 Polynomial Time Inference from Positive Data

Inference machine is an effective procedure which requires inputs from time to time and produces outputs from time to time. Let $s = s_1, s_2, \ldots$ be an arbitrary infinite sequence, and let $g = g_1, g_2, \ldots$ be a sequence of outputs produced by an inference machine M when inputs in s are successively given to M on request. Then we say that M on input s converges to g_0 iff g is a finite sequence ending with g_0 or all but finitely many elements of g are equal to g_0.

Let $L = L_1, L_2, \ldots$ be an indexed family of recursive languages, and let $s = s_1, s_2, \ldots$ be an arbitrary enumeration of some language L_i. Then we say that a machine M infers L from positive data if M on input s converges to an index j with $L_j = L_i$. We say that a family L is inferrable from positive data if there exists a machine which infers L from positive data.

Theorem 1. [Angluin, 2] If a class $L = L_1, L_2, \ldots$ satisfies the following condition, then L is inferrable from positive data.

Condition: For any non-empty finite set S of strings, the set $\{ L \mid S \subseteq L, L = L_i \text{ for some index } i\}$ has finite cardinality.

Lemma 2. [Angluin, 2] The class of pattern languages satisfies Condition of Theorem 1.

Hereafter we omit the phrase "from positive data", hence for example "inference" means "inference from positive data."

An inference by a machine M is consistent iff a language L_{g_i} contains all inputs given so far whenever M produces output g_i. An inference is conservative iff an output g_i from M is never changed unless L_{g_i} fails to contain some of the inputs. These two properties natural and valuable in inference problem. It is,

however, known that inferrability does not always mean consistency and conservativeness [Angluin, 2].

A class L is polynomial time inferrable iff there exists an inference machine M which infers L consistently and conservatively, and requests a new input in polynomial time (with respect to the length of the inputs read so far) after the last input has been received.

MINL calculation for a class $L = L_1, L_2, \ldots$ is defined by Angluin [2] as follows:

MINL(S) = "Given non-empty finite set S of strings, find an index i such that $S \subseteq L_i$ and for no index j, $S \subseteq L_j \subsetneq L_i$."

The following theorem shows the importance of MINL calculation.

Theorem 2. [Angluin, 2] If a class $L = L_1, L_2, \ldots$ satisfies Condition of Theorem 1 and MINL for L is computable, then the procedure Q below infers L consistently and conservatively.

```
procedure Q;
  begin
    g_1 := "none" ;  S := ∅ ;
    for each input s_i do
       begin
          S := S ∪ {s_i} ;
          if s_i ∈ L_{g_i} then
                g_{i+1} := g_i
          else
             begin
                g_{i+1} := MINL(S) ;
                output g_{i+1}
             end
       end
  end
```

Corollary 1. If a class $L = L_1, L_2, \ldots$ satisfies Condition of Theorem 1, and the membership decision and MINL calculation for L are computable in polynomial time, then the class L is polynomial time inferrable (from positive data).

Angluin [2] showed that the membership decision of pattern languages is NP-complete and ℓ-MINL calculation, a special case of MINL for pattern languages, is NP-hard.

ℓ-MINL(S) = "Given non-empty finite set S of strings, find a pattern of maximum possible length which represents a minimal pattern language containing S."

The following summarize the results of our previous study.

Lemma 3. For any regular pattern p and any string w, whether $w \in L(p)$ is decidable in $O(|p|+|w|)$ time.

Theorem 3. The following procedure computes ℓ-MINL(S) for regular pattern languages in $O(m^2n)$ time, where $m = \max\{|w|; w \in S\}$, $n = \mathrm{card}(S)$, and $w = a_1 \ldots a_k$ ($a_i \in \Sigma$) is one of the shortest strings in S.

```
begin
   p1 := x1...xk ;
   for i := 1 to k do
      begin
         q := pi[ai/xi] ;
         if S ⊆ L(q) then pi+1 := q
                     else pi+1 := pi ;
      end ;
   return pk+1
end
```

Theorem 4. The classes of regular pattern languages is polynomial time inferrable (from positive data).

3. Some Problems on ℓ-MINL Calculation for Regular Pattern Languages

There are some difficulties in the ℓ-MINL calculation when the polynomial time inference of regular pattern languages is applied to practical use. The main reasons are in

1) restriction on the length of pattern, and
2) prohibition against substituting empty string for any variable.

We present some examples to explain these problems.

Example 1. Let S be the set {ABCdeFGh, ABCiFGjk}. Then every answer of ℓ-MINL is eight symbols long because the length of the shortest words in S is eight. Let p be any pattern of the form p_1FGp_2, where p_1 and p_2 are any regular patterns. Assume $S \subseteq L(p)$. Then, clearly, ABCi $\in L(p_1)$ and h $\in L(p_2)$, therefore

$$|p| = |p_1| + |FG| + |p_2| \leq 4 + 2 + 1 = 7 < 8.$$

Hence the string "FG" does not appear in any answer of ℓ-MINL(S). However the pattern $q = ABCx_1FGx_2$ is a possible answer of MINL(S).

Thus MINL(S) may have an answer which contains more constant symbols than any answer of ℓ-MINL(S).

Example 2. Let S = {aBcdf, GHcdBiii}. Then both patterns

$p_1 = x_1Bx_2x_3x_4$ and

$p_2 = x_1x_2cdx_3$

are correct answer of ℓ-MINL(S). Our ℓ-MINL algorithm of Theorem 2 returns p_1 for S. If we change the order of substitutions in the algorithm, we can get p_2 as the answer of ℓ-MINL(S).

Example 3. Let S = {ABC, AC}. Then MINL(S) does not have any answer containing both symbols A and C because we can not substitute empty string for any variable.

To solve these problems, we extend the definition of pattern languages to allow erasing substitutions.

4. Extension of Pattern Languages

We give new definitions of pattern languages to allow substitutions to erase variables and we show some their properties. The definitions of patterns and regular patterns are the same ones as in Section 2.

A substitution is any (possibly erasing) homomorphism from P to P which maps each constant symbol to itself. A special substitution which maps each variable to empty string is denoted by c. For example, if Σ = {0, 1, 2} and X = {x, y,... }, then c(0x1y2) = 012. We define two binary relations $\leq'$ and $\equiv'$ as follows:

1) $p \leq' q$ iff $p = f(q)$ for some substitution f,

2) $p \equiv' q$ iff $p \leq' q$ and $q \leq' p$.

The language of a pattern p, denoted by $L(p)$, is the set $\{ w \in \Sigma^* \mid w \leq' p \}$. Hereafter we use the term "pattern languages" in the sense just defined above.

Proposition 1.

1) $p \leq' q \Longrightarrow L(p) \subseteq L(q)$

2) $p \equiv' q \Longrightarrow L(p) = L(q)$

We say that a pattern $\hat{p}$ is in canonical form iff

$\hat{p} \equiv' q \Longrightarrow |\hat{p}| \leq |q|$ for any pattern q, and

$\hat{p}$ contains exactly k variables $x_1, x_2, \ldots, x_k$ for some integer k and the leftmost occurrence of x_i is to the left of the leftmost occurrence of x_{i+1} for $i = 1, \ldots, k-1$.

Theorem 5. There exists a unique canonical pattern $\hat{p}$ equivalent ($\equiv'$) to p for any regular pattern p.

Proof. Let $p = w_0 x_{11} \cdots x_{1i_1} w_1 x_{21} \cdots w_{n-1} x_{n1} \cdots x_{ni_n} w_n$ ($w_0, w_n \in \Sigma^*$, $w_i \in \Sigma^+$ ($i=1,\ldots,n-1$)). Then $\hat{p} = w_0 x_1 w_1 x_2 \cdots w_{n-1} x_n w_n$ is in canonical form and $\hat{p} \equiv' p$. Any pattern equivalent to p is of the form $w_0 v_1 w_1 v_2 \cdots w_{n-1} v_n w_n$ ($v_i \in X^+$). Therefore the uniqueness of such canonical pattern is obvious. □

Lemma 4. If $\hat{p}$ is a canonical regular pattern, then

$$|\hat{p}| \leq 2|c(\hat{p})| + 1.$$

Theorem 6. The class of (extended) regular pattern languages satisfies Condition of Theorem 1 and it is inferrable from positive data.

Proof. Let $S \subseteq \Sigma^*$ be any non-empty finite set of strings and let w be one of the shortest strings in S. Assume $S \subseteq L(\hat{p})$, where $\hat{p}$ is any canonical regular pattern. Then $|w| \geq |c(\hat{p})|$ because $w \in L(\hat{p})$. By Lemma 4, $|\hat{p}| \leq 2|c(\hat{p})| + 1 \leq 2|w| + 1$. Therefore the number of such patterns $\hat{p}$ is finite. □

Theorem 7. For any regular pattern p and any string w, whether $w \in L(p)$ is decided in $O(|p|+|w|)$ time.

Proof. We can construct a deterministic finite automaton recognizing L(p) in $O(|p|)$ time by using the method of pattern matching machines [Aho, et al., 1]. □

5. MINL calculation for Regular Pattern Languages

To show polynomial time inferrability of regular pattern languages, we need discussions on MINL calculation. In this section we also refer to the relation between MINL calculation and the longest common subsequence (LCS for short) problem.

First we give some definitions on subsequences:

1) For any strings $w = a_1 \ldots a_k$ $(a_i \in \Sigma)$ and $s_i \in \Sigma^*$,
$s \leq w$ (or $w \geq s$) iff $s = a_{i_1} \ldots a_{i_m}$ $(1 \leq i_1 < \ldots < i_m \leq k)$.
We say that s is a subsequence of w (or w is a supersequence of s) if $s \leq w$ (or $w \geq s$).

2) The set of common subsequences to a set S of strings is
$CS(S) = \{ s \in \Sigma^* \mid s \leq w \text{ for any string } w \in S \}$.

3) The set of maximal common subsequences to S is
$MCS(S) = \{ s \in CS(S) \mid s = s' \text{ or } s \not\leq s' \text{ for any } s' \in CS(S) \}$.

4) The set of the longest common subsequences to S is
$LCS(S) = \{ s \in CS(S) \mid |s| \geq |s'| \text{ for any } s' \in CS(S) \}$.

Proposition 2.

1) $w \in L(p) \Longrightarrow w \geq c(p)$
2) $L(p) \subseteq L(q) \Longrightarrow c(p) \geq c(q)$
3) $L(p) = L(q) \Longrightarrow c(p) = c(q)$
4) $S \subseteq L(p) \Longrightarrow c(p) \in CS(S)$

We need three notations in the discussions below:

1) For any string $w = a_1 \ldots a_n$ and any integers i and j,

$$w\langle i:j\rangle = \begin{cases} a_i \ldots a_j & (\text{if } 1 \leq i \leq j \leq |w|) \\ \varepsilon & (\text{otherwise}), \text{ and} \end{cases}$$

$w\langle i\rangle = a_i$ $(i = 1, \ldots |w|)$.

2) For any symbol a and any integer i,

$$a^i = \begin{cases} \varepsilon & (\text{if } i \le 0) \\ aa^{i-1} & (\text{otherwise}). \end{cases}$$

3) For any variables $v_1,\ldots,v_k \in X$ and any constant strings $w_1,\ldots,w_k \in \Sigma^*$, $[w_1/v_1,\ldots,w_k/v_k]$ denotes the substitution which maps each variable v_i to w_i and every other variable to itself.

Theorem 8. Let p and q be any regular patterns and $\mathrm{card}(\Sigma) \ge 3$. Then $L(p) \subseteq L(q)$ implies $p \le' q$.

Proof. We may assume, without loss of generality, that p and q are canonical regular patterns. We also assume $\mathrm{card}(\Sigma) \ge 3$, $L(p) \subseteq L(q)$, but $p \not\le' q$. Let $q = w_0x_1w_1\ldots w_{n-1}x_nw_n$, where $w_0, w_n \in \Sigma^*$, and $w_i \in \Sigma^+$ $(i=1,\ldots,n-1)$. Since c(p) is a supersequence of c(q) and $p \not\le' q$, there exist integers i, j, and k such that

$0 \le i \le n$, $1 \le j < k \le |p|$, and
$p = p\langle 1{:}j\rangle\ p\langle j+1{:}k-1\rangle\ p\langle k{:}|p|\rangle$, where
$c(p\langle 1{:}j\rangle) \in L(w_0x_1\ldots x_{i-1}w_{i-1})$,
$c(p\langle 1{:}j'\rangle) \notin L(w_0x_1\ldots x_{i-1}w_{i-1})$ for any integer $j' < j$,
$p\langle j+1{:}k-1\rangle \ne rw_ir'$ for any patterns r and r',
$c(p\langle k{:}|p|\rangle) \in L(w_{i+1}x_{i+2}\ldots x_nw_n)$, and
$c(p\langle k'{:}|p|\rangle) \notin L(w_{i+1}x_{i+2}\ldots x_nw_n)$ for any integer $k' > k$.

Let $p_1 = p\langle 1{:}j\rangle$, $p_2 = p\langle j+1{:}k-1\rangle$, and $p_3 = p\langle k{:}|p|\rangle$. Then $L(p_2) \subseteq L(x_iw_ix_{i+1})$ because $L(p) \subseteq L(q)$ and $c(p_1)p_2c(p_3) \le' p$. Let a be any constant symbol except $w_i\langle 1\rangle$ and $w_i\langle |w_i|\rangle$ and let $v_1, \ldots, v_m$ be all variables in p_2. Then $p[a^{|w_i|}/v_1,\ldots,a^{|w_i|}/v_m] \in L(p_2) - L(x_iw_ix_{i+1})$. This contradicts $L(p_2) \subseteq L(x_iw_ix_{i+1})$. □

The following lemma says that the condition $\mathrm{card}(\Sigma) \ge 3$ is necessary in Theorem 8.

Lemma 5. When $\mathrm{card}(\Sigma) = 2$, there exist regular patterns p and q such that $L(p) \subseteq L(q)$, $p \not\le' q$, and $q \not\le' p$.

Proof. Let $\Sigma = \{0,1\}$, $p = x_101x_20x_3$, and $q = x_10x_210x_3$. Then, clearly, $p \not\le' q$, $q \not\le' p$, but $L(p) = L(q)$. □

Hereafter we assume that the constants alphabet Σ contains at least three symbols.

Theorem 9. For any maximal common subsequence $s \in MCS(S)$, there exists an answer p of MINL(S) for regular pattern languages such that $c(p) = s$.

Proof. Let $s = a_1 \ldots a_k \in MCS(S)$. Then the pattern q_{k+1} defined as follows is an answer of MINL(S):

$$q_i := \begin{cases} x_1 a_1 \ldots a_k x_{k+1} & (i=0) \\ \text{if } S \subseteq L(q_{i-1}[\varepsilon/x_i]) \text{ then } q_{i-1}[\varepsilon/x_i] \text{ else } q_{i-1} & (i=1,\ldots,k+1). \end{cases}$$

We must show that $L(q_{k+1})$ is a minimal regular pattern language containing S. Assume that there exists a regular pattern q' such that $S \subseteq L(q') \subsetneq L(q_{k+1})$. Then $c(q') \geq c(q_{k+1}) = s$. Since s is a maximal common subsequence to S, $c(q') = c(q_{k+1}) = s$. By Theorem 8, $q' \leq' q_{k+1}$ and $q' \not\equiv' q_{k+1}$. There exists a substitution f which maps q_{k+1} to q'. The substitution f maps at least one variable to empty string because $q' \not\equiv' q_{k+1}$. Let j be an integer such that x_j appears in q_{k+1}, $f(x_j) = \varepsilon$, and $f(x_{j'}) = x_{j'}$, for any integer $j' < j$. Then $q' \leq' q_{j-1}[\varepsilon/x_j]$. Therefore $S \subseteq q_{j-1}[\varepsilon/x_j]$ and $q_j = q_{j-1}[\varepsilon/x_j]$. Hence the variable x_j can not appear in q_{k+1}. This contradicts the selection of j. □

Here we should note that we can get an answer of MINL(S) in $O(m^2n)$ time from any maximal common sequence to a set S of strings, where $m = \max\{|w|;\ w \in S\}$ and $n = \text{card}(S)$.

We may prefer the longest common subsequences to the maximal common subsequence. However the problem to find one of the longest common subsequences to a set of strings is known to be NP-complete [Maier, 8]. Therefore finding an answer of MINL(S) containing constants as many as possible does not seem to be done in polynomial time. To find an answer of MINL(S) for regular pattern languages, is it necessary to select one of the maximal common subsequences to S? The following theorem asserts that it is not the case.

Theorem 10. There exists an answer p of MINL(S) for regular pattern languages such that $c(p) \notin MCS(S)$ for some set S of strings.

Proof. Let S = {01020, 0212}. Then $02 \notin MCS(S)$ because $012 \in CS(S)$. However the pattern $p = x_1 02 x_2$ represents a minimal regular pattern language containing S. □

In the proof of Theorem 10, the pattern $q = 0x_1 1x_2 2x_3$ is a possible answer of MINL(S) and $c(q) = 012 \in LCS(S)$. In some cases q is not always better answer of MINL(S) than p because the pattern p contains a longer constant string "02" than q. Finally, from this obsevation, we get a MINL algorithm by using a method to find common strings in length decreasing order. The correctness is easily shown by Theorem 8 and the computing time is $O(m^4 n)$, where $m = \max\{|w|; w \in S\}$ and n = card(S).

In our MINL algorithm we use some notations for simplicity:

Let $\sigma = w_1, \ldots, w_n$ be a sequence of strings. The notation $L(\sigma)$ denotes the regular pattern language $L(x_1 w_1 x_2 \ldots w_n x_{n+1})$, $|\sigma|$ denotes the number of strings in σ, and $\|\sigma\|$ denotes the sum of lengths of strings in σ.

```
Procedure MINL(S);

  (* Input S: non-empty finite set of strings *)
  (* Output p: a pattern representing a minimal
               (extended) regular pattern language containing S *)

   begin
     s := one of the shortest strings in S ;
     σ := ε;          (* sequence of common strings *)
     n := |s| ;       (* length of candidate string *)
     while n > 0 do begin
       for i := 1 to |s| - n + 1 do
more:    for j := 0 to |σ| do
           if S ⊆ L(σ<1:j>,a<i:i+n-1>,σ<j+1:|σ|>)
             then begin
                    σ := σ<1:j>,a<i:i+n-1>,σ<j+1:|σ|>) ;
                    go to more
             end ;
       n := min(|s|-‖σ‖, n-1)
     end ;
     p := x1σ<1>x2...σ<|σ|>x|σ|+1 ;
     if S ⊆ L(p[ε/x1]) then p := p[ε/x1] ;
     if S ⊆ L(p[ε/x|σ|+1]) then p := p[ε/x|σ|+1] ;
     return p ;
   end
```

Theorem 11. The class of (extended) regular pattern languages is polynomial time inferrable (from positive data).

6. Concluding Remarks

We have discussed polynomial time inference for the class of the extended regular pattern languages and we have seen that MINL calculation for the class plays an important role in inference from positive data. We have also discussed the relation between the MINL calculation and the longest common subsequence problem.

It should be noticed that our MINL algorithm for the extended regular pattern languages is consistent to the NP-completeness of the LCS problem. The MINL algorithm finds common strings to a set in length decreasing order. It should also be noticed that our method in the algorithm is natural.

Since our evaluation of the time complexity is not so acute, the exponent of the maximum length of strings might be reduced. The MINL algorithm is originally designed for the learning data entry system, and it should have other practical applications. A little modification may be needed for some problems.

ACKNOWLEDGMENTS

The author wishes to acknowledge Professor S. Arikawa for his helpful suggestions and encouragement. He would also like to thank Mr. S. Miyano for his useful comments in the course of starting this study.

REFERENCES

[1] Aho, A.V., Hopcroft, J.E. and Ullman, J.D. (1974), The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass.

[2] Angluin, D. (1979), Finding Patterns Common to a Set of Strings, in Proceedings, 11th Annual ACM Symposium on Theory of Computing, pp. 130-141.

[3] Angluin, D. (1980), Inductive Inference of Formal Languages from Positive Data, Inform. Contr. 45, 117-135.

[4] Arikawa, S. (1981), A personal communication.

[5] Gold, E.M. (1967), Language Identification in the Limit, Inform. Contr. 10, 447-474.

[6] Hirschberg, D.S. (1977), Algorithms for the Longest Common Subsequence Problem, JACM 24, 664-675

[7] Hopcroft, J.E. and Ullman, J.D. (1969), Formal Languages and their Relation to Automata, Addison-Wesley, Reading, Mass.

[8] Maier, D. (1978), The Complexity of Some Problems on Subsequences and Supersequences, JACM 25, 322-336.

[9] Shinohara, T. (1982), Polynomial Time Inference of Pattern Languages and its Application, in Proceedings, 7th IBM Symposium on Mathematical Foundation of Computer Science.

[10] Wagner, R.A., and Fischer, M.J. (1974), The string-to-string Correction Problem, JACM 21, 168-73.